SPERRY UNIVAC

P.O. BOX 500
BLUE BELL, PENNSYLVANIA 19422
TELEPHONE (215) 542-4011

Hella

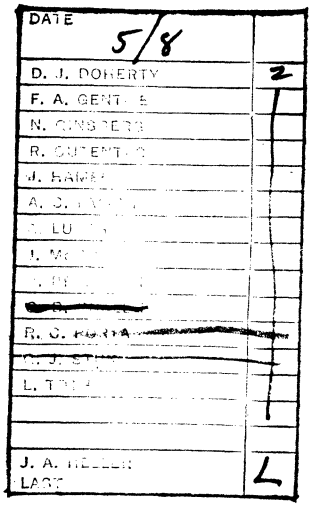
f.y.i. Larry

April 10, 1979

L. Bernstein
Director, Facilities Development Center
Bell Telephone Laboratories
6 Corporate Place
Piscataway, New Jersey 08854

Dear Larry:

The Sperry Univac 1100/60 Series of computer systems will be announced at the National Computer Conference in New York on June 5, 1979. This system offers many features and benefits which will improve the economics of BISCUS/FACS and BISCUS/PREMIS and speed deployment.



The 1100/60 is the newest member of the Sperry Univac 1100 Series product line. It will serve as the entry level model offering competitive cost/performance with the IBM "E" series. It utilizes many technological innovations in its architecture, while remaining compatible with other members in the 1100 Series. The general characteristics of the 1100/60 are listed below and in the attached Hardware overview.

- This system spans the performance range from below the recently announced IBM 4341 up to the performance of the IBM 3032. Pricing of the system will be competitive with the new price/performance curve of the IBM 4331 and 4341.

- The 1100/60 system offers six distinct processor performance levels. These levels are listed below:

  - Model C1 - unit processor
  - Model C2 - unit processor with extended instruction set
  - Model H1 - unit processor with cache
  - Model H2 - unit processor with cache and extended instruction set
  - Model H1 - multiprocessor (2X)
  - Model H2 - multiprocessor (2X)

  In addition, expansion to 3X and 4X Multiprocessor systems are planned but are not being announced at this time. This future announcement will allow system performance in the range of the IBM 3033.

- Up to 1048M words (4M bytes) of memory per processor is configurable in the system. Future increases in memory capacity are planned.

- The 1100/60 Systems' performance can be expanded without equipment exchange to an 1100/62 multiprocessor with over 4.5 times the processor performance of the C1 Model.

- Through the use of LSI technology and improved packaging techniques, the entire system (CPU, IOU, cache, memory) can be contained in a single cabinet (3X5 feet). Corresponding reductions in power and environmental requirements are also provided.

- The 1100/60 System, though using many innovative architectural techniques, is compatible with the full 1100 Series product line. Application code developed for the 1100/80 System will also run on the 1100/60.

- The 1100/60 System will be the first unbundled Sperry Univac 1100 Series product.

- The 1100/60 System is the first 1100 Series product to include instructions specifically provided for UNIX/1100. These instructions will improve performance for character reference and dereference operations.

- The 1100/60 System will allow simultaneous use of BICS and UNIX/1100 providing an extremely attractive development environment.

Several new peripherals will also be announced at this time. Their highlights are given below:

- The 8470 Disk and its associated controls will provide 635 M Byte disk drives competitive with the new IBM 3570 disk family. These drives in fact offer larger capacity with an earlier delivery date.

- The new Uniservo 22/24 tape systems provide low cost tape handling for those applications not requiring 6250BPI.

- An improved low cost 1200 LPM printer is available.

These systems provide smaller entry levels for the OTC's, with a more flexible growth and entity sizing for either 1100/60's or the larger 1100/80 Product line. The improved economics will greatly reduce PREMIS cost and improve savings. BISCUS/FACS can be modularly implemented without requiring large initial capital investments in hardware.

Deliveries of this new Series will begin late this year. We would like to invite yourself and members of your staff to visit our Roseville Development Center for detailed presentations on this new product at your earliest convenience.

* UNIX is a trademark of Bell Telephone Laboratories

For your planning purposes, we expect that the Univac 1100/60 will have an improved price performance of 40% over the Univac 1100/80 and we also expect that we will continue to be competitive on the Univac 1100/80 as other announcements are made. We are preparing more definitive cost and performance information of the 1100/60 which we will be able to give to you next week. In addition a proposal will be prepared to enable Bell Telephone Laboratories to install one of the first systems for development activities. This proposal including firm pricing will be available in approximately 30 days.

As we have shown, Sperry Univac is fully competitive with current competitive product offerings and will continue to be through technological and competitive leadership.

Sperry Univac is fully committed to the successful implementation of BISCUS/FACS and BISCUS/PREMIS and we are available to provide assistance in this program.

Branch Manager
Bell Branch

CC J. J. Yostpille
J. M. Nervik

PLANNING PRICING COMPARISON

Central Complex

	Cost Millions	Instruction Time	Relative Cost
1110 2X2	$5.4	600 NS	3.24
1100/82	$3.1	335 NS	1.04
1100/60	$ .4	1470 NS	.58

Memory

1 Million Words (4 Mega bytes)

1110	$1,992,000
1100/82	515,000
1100/62	100,000

Mass Storage

Dual Control & 8 Drives

		Cost	Storage	Cost/Storage
1110	8440	382K	160 M WDS	2.39
1100/80	8433	485K	272 M WDS	1.78
	8450	440K	432 M WDS	1.02
1100/60	8470	364K	1067 M WDS	.34

4.0 HARDWARE OVERVIEW

The system complex is based on 1100/80 technology utilizing 10K ECL and TTL circuitry, microprocessors and multi-layer packaging techniques. The CPU, IOU, cache-buffer, support controller and main storage unit are contained in one cabinet including power. This approach plus paralleling microprocessors provides improved cost/performance characteristics compared to the current 1100/80. Main memory within the basic 1x1 complex starts at 262K words, expandable to 1,048K words within the single cabinet.

A System Support Processor (SSP) interfaces to the basic system and interfaces directly with the main storage. Several 1100 functions are performed by the SSP including systems partitioning, system console, initial load, auto recovery and maintenance check of the system.

- System Cabinetry - As opposed to most recent 1100 Systems, the 1100/60 Processing Complex is housed in a single UPSIII cabinet. This reduces cost associated with providing separate cabinets and power sources for each Central Complex component. Cost and performance are also enhanced by the reduction of system interconnects associated with this concept.

- System Configuration - the 1100/60 Processing Complex is composed of a Central Processing Unit (CPU), Input/Output Unit (IOU), System Interface Unit (SIU), and Main Storage Unit (MSU). The minimum system is a 1x1 with 262K words of storage and no SIU. It may be expanded by the addition of 8K words of SIU, and up to 1048K words of MSU. The maximum system at initial announcement will be a 2x2 with 16K words of SIU and 2096K words of storage. Further expansion to a 4x4 with 32K words of SIU and 8 million words of storage will be announced approximately 1 year later. At that time, the maximum storage configuration of the 1x and 2x systems will also be increased to 8 million words. This will be accomplished by utilizing one or two separate cabinets to house storage, each with up to 4 million words of storage. All 3x and 4x systems will require these separate (external) storage cabinets.

- Central Processing Unit - The 1100/60 CPU design is based on ECL and micro-programming technology. Paralleled Arithmetic Logic Unit (ALU) chips are used to gain design flexibility, reduced cost, reduced size, and reliability. The 1100/60 instruction set is micro-programmed utilizing a separate random access control store as the storage media. Reliability is enhanced through use of duplex checking, extensive parity generation and checking, control store error correction and instruction retry. Packaging techniques are the same as those used for the 1100/8

- Input/Output Unit - IOU design is based on concepts utilized for the 1100/80. The minimum IOU configuration contains 1 Block Mux Channel Module and 1 Word Channel Module (4 word channels). Expansion capabilities exist to increase this to 2 Block Mux Channel Modules and 3 Word Channel Modules (12 Word Channels),or 3 Block Mux Channel Modules and 2 Word Channel Modules (8 word channels). The technology used is T^2L.

- Main Storage Unit - The 1100/60 MSU uses a 16K MOS chip to achieve a maximum of 1M words in a storage module. The storage array card is the same as that used in the 7037 Storage Unit (1100/80), with the exception being the specification of faster storage chips. The MSU utilizes a five port MMA to interface with a maximum of two CPU's or SIU's, two IOU's and 1 Support Controller.

  Single bit error correction with double bit error detection is the error detection/correction technique utilized.

- System Interface Unit - The SIU is a high speed 8K buffer memory dedicated to interfacing its assigned CPU with Main Storage. In 1100/60 Systems with SIU, the cabling between processor and Main Store (non-SIU systems) is replaced with an SIU and associative cabling, thus directing CPU storage references to the SIU. This allows for the configuring of an 1100/60 System with or without SIU without duplicating interfaces.

- Support Controller (SC) - The SC provides the system interface between the SSP and each module. Up to two SSP's can be interfaced per SC.

- System Support Processor - The SSP is a product line UTS 700/BC/7 Processor configured with supportive diskette storage, Maintenance Interface Adapter, System Interface Adapter and 65K bytes of storage. A U200 provides the System Console function. The SSP interfaces directly with main storage through an MMA port and with the CPU, SIU, and IOU through the SC scan set interface.

  Several 1100 functions previously performed via separate hardware components have been replaced by the SSP application. They are:

  - System Partitioning
  - System Console
  - Maintenance Panel
  - Processor Controls
  - Maintenance Processor

Other functions supported by the SSP are:

- Initial Load
- Auto Recovery
- Control Store Loading
- Trace Support
- Error Analysis
- On-Line/Off-Line Maintenance
- Performance Monitoring

- Field Instruction Set - The field instructions are 18 new COBOL oriented instructions designed to enhance the byte/character manipulate capabilities of 1100/60 Systems. For multi-processor systems, one field instruction feature is required for each central processor if the enhanced instruction set is desired. The appropriate compilers will be modified to take advantage of the new instructions if they exist in an 1100/60 System.

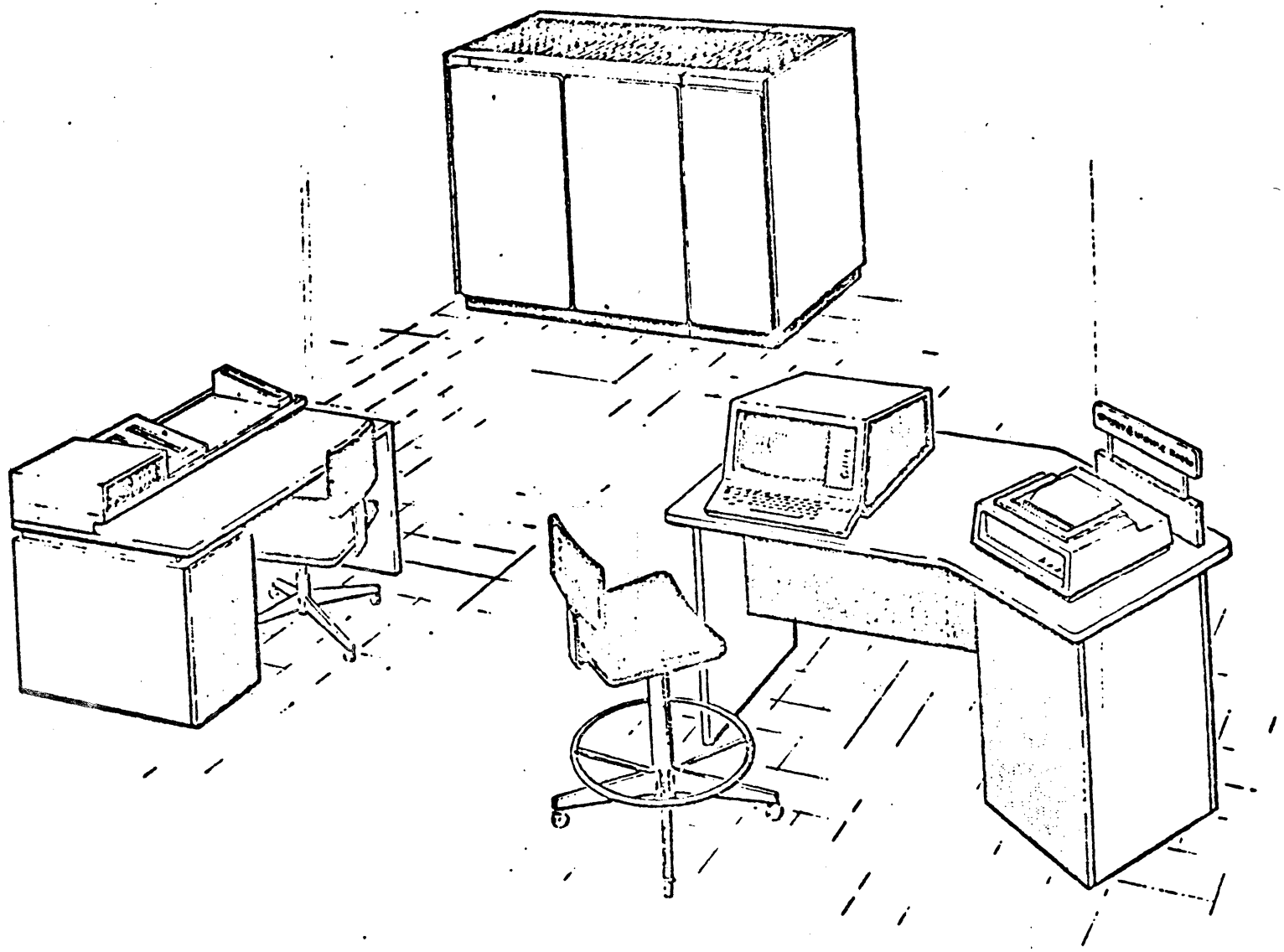
- Performance Monitor - This feature provides a CPU with the capability to collect system profile hardware data and software performance data. The hardware related data will provide utilization of individual hardware modules such as processor busy and individual I/O Channel activities, as well as interdependencies between them (e.g. CPU idle and a given I/O Module active). The software related data will provide system or user software state information. This feature may be used in conjunction with the Software Instrumentation Package (SIP), although SIP need not be used to collect the profile data. If the performance monitor feature is selected, one feature will be required in each 1100/60 processing complex.

- System Partitioning - Partitioning of central complex components (CPU, IOU, SIU, MSU) is a standard feature of the system. It is accomplished by the SSP Software enabling and disabling hardware interfaces in the system.

  Partitioning of subsystems is accomplished by adding optional partitioning features to the system. These will also be controlled by the SSP software, and will enable/disable SPI parts, or switch channels to byte subsystems by remotely controlling a Byte Channel Transfer Switch.

  Partitioning of 1100/60 Systems will be functionally equivalent to 1100/80 Systems with the combination of the TU, SAU and BCTS.

CENTRAL COMPLEX



Introduced last month at the NCC, the new processor contains a "phantom branch" mechanism to avoid lost cycles in its overlapped structure . . . and a good deal more.

THE MICROARCHITECTURE OF UNIVAC'S 1100/60

Several alternative LSI implementation approaches are available as potential candidates for use in a processor. These include custom, semicustom, gate array, hybrid (multichip), and multimicroprocessor. With respect to the objective of implementing a medium performance LSI version of an existing processor macroarchitecture, each has drawbacks which include various combinations of high design costs, long design time, part type proliferation, low speed, inefficient use of chip area, high cost, and pin count limitation.

A multimicroprocessor design technique has been implemented in the Sperry Univac 1100/60. A significant feature of this technique is that it allows the implementation of existing macroarchitectures without the software base. This software compatability is achieved with improvements in cost/performance, and allows a large amount of duplicated logic to be economically incorporated into the system to achieve a high degree of fault detection.

It was recognized from the beginning that simply ganging sufficient microprocessors to form a full-word arithmetic logic unit would fall short of the performance target. Many possibilities of achieving higher performance were investigated. The approach that appeared most promising was to provide multiple microinstruction execution units that would concurrently execute parts of a macroinstruction. Thus, each macroinstruction would be decomposed into a set of atomic operations. Atomic operations that can be executed concurrently are identified and are executed in parallel on separate microinstruction execution units. To increase performance still further, execution of microinstructions is overlapped and a unique branching scheme is used to avoid lost microcycles due to conditional branching in the microcode.

To reduce costs, the inputs and outputs of the microinstruction execution units are bussed. This eliminates the need for logic to steer the information in and out of these execution units. Investigations indicated that the use of common input and output buses would not significantly impact performance. In addition to saving gating logic, the input bussing allows the use of a single shifter and the

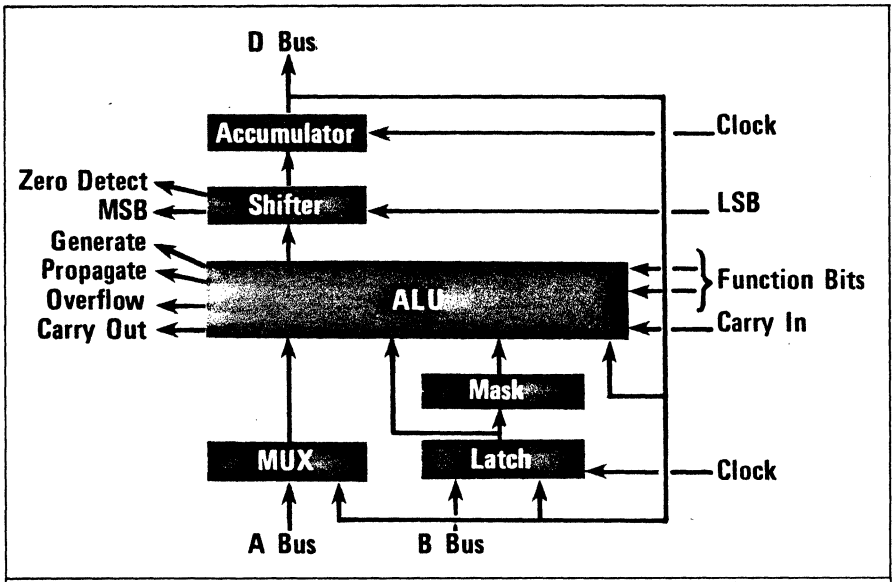


Fig. 1. 10800 Microprocessor

output bussing makes it economically viable to duplicate the microinstruction execution units and compare results at a single point.

The fastest available microprocessor slice, the Motorola 10800, was selected as the LSI building block for the 1100/60. It is a 4-bit slice using 10K ECL technology. None of the companion chips designed specifically to be used with the 10800 (e.g., control chip) were used. The surrounding logic is composed of conventional MECL 10K components with heavy emphasis on using four- and eight-input multiplexor chips and various PROM's and RAM's.

Fig. 1 is a simplified diagram of the 10800 showing the paths that are actually used in the 1100/60. The basic instruction repertoire consists of add, subtract, complement, shift one bit, AND, OR, exclusive OR, and NOT. More complex functions such as multiply and divide are achieved by microprogramming. The mask network shown in Fig. 1 allows a Boolean function to precede an arithmetic function during the same microcycle; this capability is used heavily to attain speed. The constraint of shifting only one bit per cycle is a severe one and necessitates a high-speed shifter. Shifting is a pin-limited function; thus it is a general problem with all bit-sliced microprocessors. Two notable differences between the 10800 microprocessor slice and a more conventional ALU slice are the inclusion of the latch on the B bus and the internal accumulator. The A bus does not have a latch, necessitating an external register composed of ECL flip-flops. The D bus can be disabled by a function bit so that a wired OR can be used on the output bus.

The timing specification of a chip as complex as this one (350 equivalent gates, LSI by any standard) is not simple. Some appreciation of the speed can be obtained from the fact that an add instruction, A+B to D, typically takes 40ns. Of more interest when emulating a 36-bit wide system is the fact that the propagate and generate signals (to carry a lookahead network) are available after only 24ns typical.

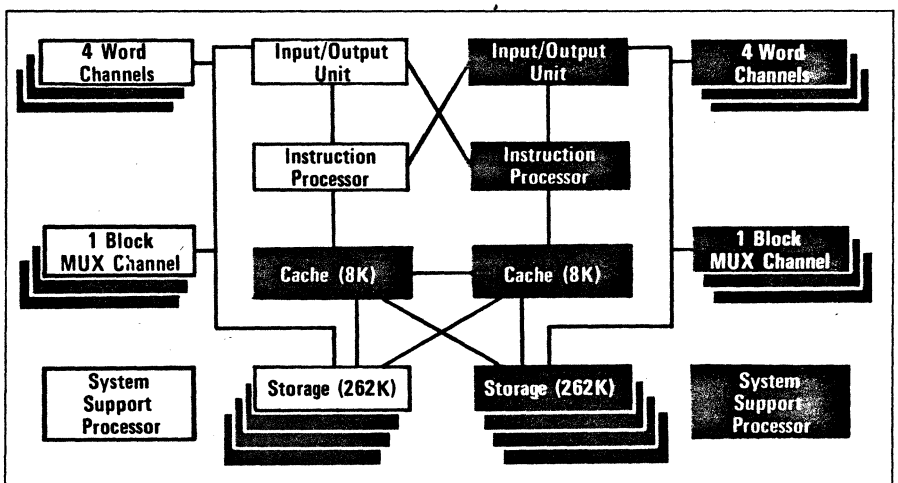


Fig. 2. White boxes show minimum configuration

1100 SERIES MACRO-ARCHITECTURE

The 1100 series architecture has been described elsewhere in some detail. The more limited purpose here is to describe enough of the 1100 Series architecture to be able to understand the multimicroprocessor implementation described below. This series, introduced with the 1107 system in 1962, is based on 36-bit instruction and operand words.

The instruction word is divided into seven fields. The *f*-field indicates the operation to be performed and specifies how the remaining fields are to be interpreted. The *j*-field either controls partial word transfers to and from storage, or it acts as an extension to the *f*-field in defining the operation to be performed.

The *a*-field selects one or more locations in the General Register Set (GRS) to provide one of the operands for each instruction. The GRS is a 128-location, high-speed random access storage in the processor. There are three primary types of registers in the GRS: the X registers are used for storage operand address indexing, the A registers are used as general purpose arithmetic registers, and the B registers are used as a special purpose working registers. An 1100 Series macroinstruction can specify operands from either one register and a main storage location, or from two registers. Most instructions operate on full or partial single-precision, 36-bit operands. However, some instructions use double precision, 72-bit operands.

The *x*-field in the instruction word specifies a GRS register to be used for storage address indexing. The X register contains two fields. One field is used as an address modifier, and the other field is used as an increment value to the modifier field. The contents of the modifier field of the X register is added to the contents of the *u*-field of the instruction to form a relative storage address. If the resultant address is less than 128, the source for both of the operands for the instruction is GRS. Otherwise it is added to a base register to form an absolute operand storage

address.

The one-bit *h*-field indicates when the X register selected by the instruction is to be automatically incremented when the instruction is executed. When $h=1$, the two fields are added and the result is used to replace the former value of the modifier field.

The *i*-field specifies indirect addressing. When this one-bit field is set, the data word read from the location specified by the address is used to form recursively another absolute operand storage address until the *i*-field of the new word is not set.

As with all 1100 Series systems delivered since 1968, the 1100/60 (see Fig. 2) is available in multiprocessor configurations. Each system support processor incorporates a maintenance processor and a console. Each instruction processor executes the 1100 Series instruction repertoire and, in addition, has new decimal and variable-length field manipulation instructions. These new instructions increase the execution speed of COBOL, and improve the execution speed of a number of other software packages. An optional 8K-word cache buffer can be supplied with each instruction processor. Each input/output unit supports up to 12-word channels and up to three block multiplexor channels, and has a direct interface main storage.

Physical packaging is very compact, with the instruction processor, I/O unit, cache buffer, and one million words of main storage fitting into a single cabinet 2.0 meters long by 0.75 meters wide. The instruction processor and cache hardware are ECL, the I/O unit is TTL, and main storage is 16K-bit MOS. Standard 1100 Series peripherals can be used with 1100/60. The speed of a unit instruction processor with a cache is about 1.3 times that of an 1108.

THE 1160's MICRO-ARCHITECTURE

With respect to the research design, the 1100/60 uses the extended instruction repertoire of the Univac 1100/80 rather than the 1108 instruction repertoire. Also, the four-base-register addressing of the 1100/80 is used rather than the two-base-register addressing of the 1108.

Seven half-word microinstruction execution units are required to minimize the address calculation time of the 1108. Because the 1100/60 has twice as many active segments, even more microinstruction units would be required. To avoid introducing a large number of microinstruction units, most of which would be used only for addressing, and to allow macroinstruction overlap, it was decided to use dedicated logic to perform address calculations. This change reduced the number of microinstruction execution units to two, and allowed overlapping at both the microinstruction and macroinstruction levels.

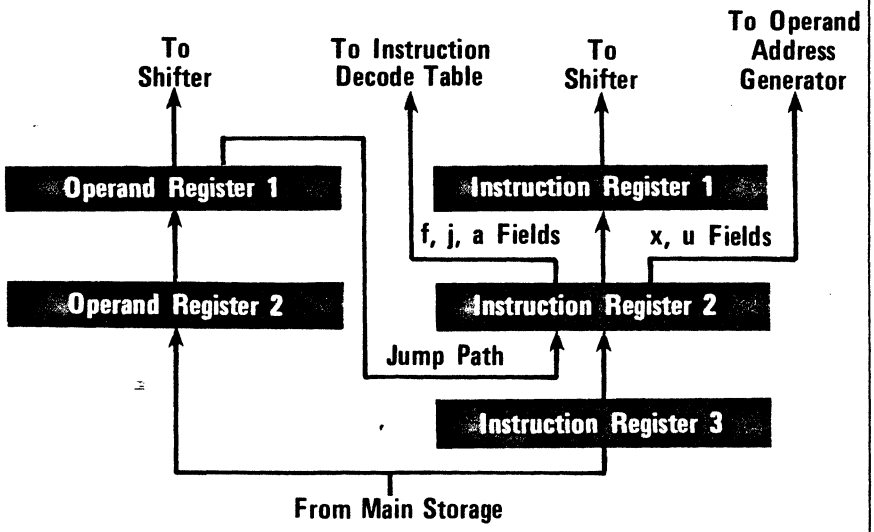


Fig. 3. Instruction and Operand Registers

The 1100/60 microarchitecture consists of a microexecution section and a storage address generation section. The microexecution section consists of two 36-bit microinstruction execution units. Each microinstruction execution unit contains a subprocessor constructed from nine 4-bit microprocessors and associated control circuitry.

The storage address generator employs four sets of base address and limits checkers which operate in parallel to allow four base additions and limits violation checks to be done simultaneously. The storage address generator operates in 116ns cycles, the same as microinstruction execution.

Operand address generation can take one or two microcycles. In the first microcycle, the *u*-field of the macroinstruction is used as a relative operand address and added to all four bases simultaneously. Within the same cycle, limits checking is performed and the proper absolute operand address is selected. If the relative address is less than 128, a designator bit is set which will be interrogated later by microde to indicate that the operand must be fetched from the GRS.

While these operations are taking place, a text of the index (x) field in the microinstruction is made. If the x-field is zero, the absolute address generated by the operation just described is used to fetch the operand. If the x-field is nonzero, the x register is read up, the contents of the modifier field are added to u to form a new relative address, and a second absolute operand address generation takes place. Two cycles are required when x is nonzero.

Instruction address generation is similar to operand address generation. The relative address of the previous instruction is kept in a holding register. A new instruction address is generated by adding one to the contents of this holding register to form a new relative instruction address, adding all four bases to the new relative address, and checking against the limits registers to select the absolute instruction address. The entire instruction address generation takes one microcycle.

Instruction and operand address generation takes place alternately in the same base adder and limits checker hardware. Requests are made until the operand and instruction registers are full as described below.

Besides the base adders and limits checkers, the storage address generator contains instructions and operand buff registers. Fig. 3 shows the operand and instruction registers. There is one storage interface port in the processor. As each request is made, the storage address generator determines whether it is a request for an instruction or for an operand. When a word comes into the processor from storage, it is routed into an operand or instruction register by control signals from the storage address generator.

Instruction Register 1 contains the macroinstruction (program instruction) currently being executed by the microcode in the processor. The next instruc-

tion to be executed is contained in Instruction Register 2. The *x*- and *u*-fields from this macroinstruction are used in the storage address generation procedure described above. While the microcode is executing the macroinstruction in Register 1, the storage address generator fetches the operand for the instruction contained in Register 2. The next macroinstruction beyond that one is contained in Register 3. Thus it is possible for three macroinstructions to be resident in the processor simultaneously.

Operand Register 1 contains the operand for the instruction currently being executed in the microcode. Operand Register 2 normally contains the operand for the next instruction. Thus operands for two instructions can be resident at one time: Operand Register 1, containing the operand for the instruction currently residing in instruction Register 1; and Operand Register 2, containing the operand for the instruction currently in Instruction Register 2.

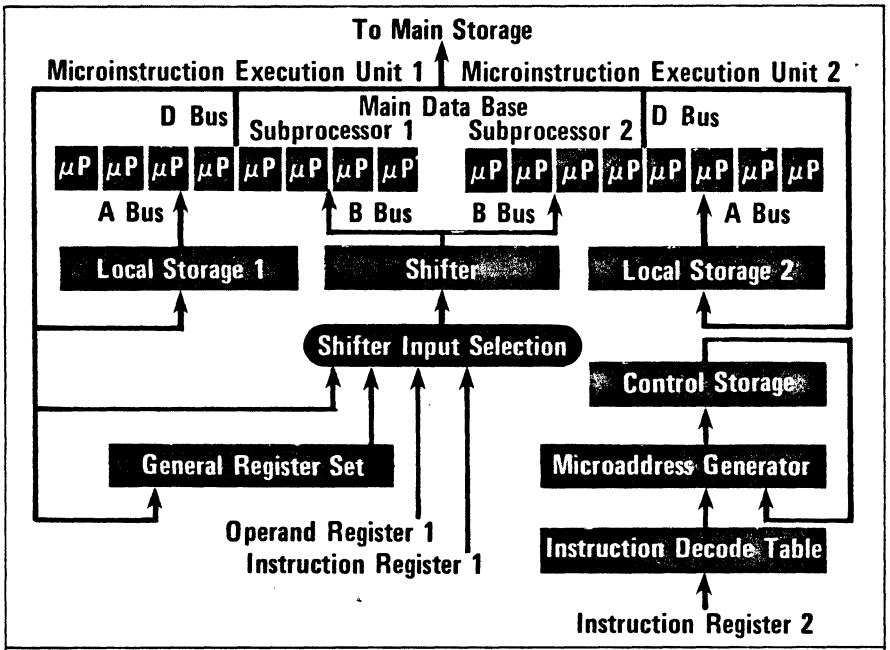


Fig. 4. Microexecution Section

MICRO-EXECUTION SECTION

When a jump or multiple operand instruction is performed, the operand registers function somewhat differently. Fig. 4 is a diagram of the microexecution section. The two subprocessors drive a single main databus which can feed main storage, the GRS, the local storage for each subprocessor, or the shifter input selector. The A bus input to each subprocessor is driven by a dedicated 256-location local storage. The B bus inputs are driven by a common 36-bit, high speed shifter.

Each 1100/60 macroinstruction is executed by a series of microinstructions. While each new macroinstruction is in Register 2 undergoing storage address generation, the *f, j,* and *a*-fields are used to generate the address of the first microinstruction of the routine which will execute the macroinstruction. This is done through the use of a 256-location, 40-bit instruction Decode Table containing one location for each macroinstruction in the 1100/60 instruction repertoire.

The output of the instruction Decode Table consists of three fields. The first field is an 11-bit *class base,* the second field is an 11-bit *instruction vector,* and the third field contains control bits. To minimize the total number of microinstructions required to execute all of the macroinstructions in the 1100/60, the execution of each macroinstruction is divided into two microroutines. The first microroutine starts at the class base address. At any point during the execution of the macroinstruction, a second microroutine may be started at the microaddress instruction vector. This allows the use of fewer total microinstructions because most macroinstructions may use a common class base microroutine and require only one or two unique microinstructions. The control bits (which can be unique to each macroinstruction) can modify the operation of the microcode in such a way that a single microroutine can be used for more than one macroinstruction, allowing further reduction in the total number of microinstructions.

When the microcode completes the execution of the current macroinstruction, the next macroinstruction to be executed is transferred from Instruction Register 2 into Instruction Register 1. Then, microroutine to execute the instruction is started at the class base address from the Instruction Decode Table.

Four 116ns microcycles are required for the completion of each microinstruction. To obtain an effective microinstruction time of 116ns, microinstructions are overlapped four deep. Fig. 5 shows the microinstruction overlap of our four typical microinstructions.

During cycle 1, the address for microinstruction is generated using fields contained in microinstruction n-1 combined with variables generated by the results of execution of microinstruction n-2. In cycle 2, microinstruction n is fetched from control store and used to control the setup of the data which will be presented to the subprocessors for use during the execution portion of the microinstruction.

During cycle 3, the execution for microinstruction n takes place. An arithmetic or logic function is performed which combines the data loaded into the A and B bus latches for each subprocessor at the end of cycle 2 with the data placed in the accumulators as a result of the execution during cycle 2 for microinstruction n-1. The results of executions are loaded into the accumulator registers toward the end of cycle 3. At the beginning of cycle 4, the contents of the accumulator register in one of the subprocessors is chosen to drive the main databus from where it may be loaded into the GRS or local store, sent to the main storage unit as data, or written into one of several other internal registers. Also during cycle 4, variables which result from the execution of microinstruction may be used to select the address for microinstruction n+2 and the functions may be executed for microinstruction n+1.

Decisions concerning microaddress generation, microfunction selection, and results storage are made under microcode control using the *logic function generator* which is a complex selector circuit controlled by the fields in the microinstructions. The fields specify the variables to be applied to the logic function generator and one of the 16 logic functions to be performed. The variables can either be static variables representing processor state information, or dynamic variables, such as subprocessor zero detect or sign bit.

PHANTOM BRANCH The logic function generator allows each subprocessor to select between two function codes during each cycle without altering the microinstruction sequence. Since this mechanism gives an apparent branch capability on each cycle without altering the microinstruction sequence, this conditional control capability is called "phantom branch." Besides creating an independent control capability, the phantom branch minimizes wasted cycles.

There are several ways in which the phantom branch can be employed to decrease the time required to execute macroinstruction. One improvement is realized by making independent real branch and phantom branch decisions during execution of any microinstruction. This capability makes it possible to shorten the path lengths required to execute a macroinstruction.

Another speed improvement due to phantom branching is that the execution functions are chosen at a later point during the execution of the microinstruction than the address is generated. This may be illustrated by observing Fig. 5.

Operation	Cycle 1	Cycle 2	Cycle 3	Cycle 4
Generate microaddress	n	n + 1	n + 2	
Set up data	n − 1	n	n + 1	n + 2
Execute microinstruction		n − 1	n	n + 1
Store results			n − 1	n

Fig. 5. Microinstruction Overlap

The address for microinstruction n must be chosen as a result of the execution for microinstruction n-2 at the end of cycle 1, but the execution functions for microinstruction n are chosen as a result of the execution microinstruction n-1 at the end of cycle 2. If a new function must be selected based on results from microinstruction n-1, it can be done a cycle earlier using the phantom branch than would be possible if the function were selected using a real branch. This allows shorter microinstruction sequences and makes possible extremely tight microinstruction loops for the performance of repetitive operations, such as multiply and divide.

Since each microinstruction contains two possible microfunctions for each subprocessor, one microinstruction is often able to do the work of two. In any given cycle, only one of the execution functions may be selected, but in another use of the same microinstruction, the other execution function may be used. This allows one microinstruction to often do the work of two and allows a reduction in the total number of microinstructions.

As an example of the ways in

which this microarchitecture may be used, the execution of an ADD macroinstruction will be described. When an ADD instruction is performed, the macroinstruction is brought into Instruction Register 3 by the storage address generator. When the macroinstruction is loaded into Instruction Register 2, an operand address generation is performed and an operand is fetched from storage, if necessary. When the previous macroinstruction completes, execution of the ADD will begin if there are no outstanding interrupt or clock update service requests. Execution begins at the class base address from the Instruction Decode Table, and at the same time the macroinstruction is transferred from Instruction Register 2 into Instruction Register 1. The next macroinstruction can then be loaded into Instruction Register 2 so that its operand address generation and fetching can be done.

The ADD macroinstruction uses the load instruction class base. The first microinstruction performs a number of functions. If the operand from storage is available, it is brought in through the shifter from Operand Register 1 and shifted if necessary as defined by the *j*-field. The operand is then masked with constants from the local storage (selected under *j*-field control) for each subprocessor and the results are placed in the accumulators during the execution portion of the first microinstruction. At the end of the setup cycle for this microinstruction, the second microinstruction is selected. A check is made to see if the operand should come from a GRS location. If so, control is transferred to a GRS read microinstruction which reads the operand from GRS rather than from storage. If the operand does not come from GRS, a check is made to see if the storage operand is resident in the processor. If it is not, control is transferred to a microinstruction which waits for the storage operand. When the operand is available in the processor, control is transferred to the ADD *instruction vector* routine. In the first microinstruction of this new routine an operand is read from GRS and added to the first operand which was previously placed in the accumulators. Then a second microinstruction stores the result of the add into the A register in the GRS and the first microinstruction or interrupt routine is selected.

An ADD macroinstruction will be performed by as few as three microinstructions in 348ns if the storage operand is available in Operand Register 1 when the execution begins. The first microinstruction brings the storage operand through the shifter and places it into the accumulators in the subprocessors. The second microinstruction adds the GDS operand to the storage operand and places the result in the accumulators. The third microinstruction stores the results back into GRS. If the first operand comes from GRS, the execution time will be increased by one microinstruction.

The multiprocessor approach is a cost-effective way to incorporate off-the-shelf LSI into a medium scale computer system while retaining software compatability. The phantom branch mechanism has been introduced as a way to substantially increase the effectiveness of a microinstruction by providing decision points late in the microcycle to select functions performed in the next microinstruction. ❋

REFERENCES

1. Borgerson, B.R.; Tjaden, G.S.; and Hanson, M.L., "Mainframe Implementation With Off-The-Shelf LSI Modules," IEEE *Computer* magazine, July 1978, pp. 42-48.

2. Borgerson, B.R.; Godfrey, M.D.; Hagerty, P.E.; and Ryken, T.R., "The Architecture of the Sperry Univac 1100 Series Systems," *Proceedings of the Sixth International Symposium on Computer Architecture,* Philadelphia, April 1979.

3. Borgerson, B.R.; Hanson, M.L.; and Hartley, P.A., "The Evolution of the Sperry Univac 1100 Series: A History, Analysis, and Projection," *Communications of the ACM,* January 1978, pp. 25-43.

Contributing to this article were the following employees of Sperry Univac: Lewis A. Boone, senior logician with design responsibility for the 1100/60 cpu; Dr. George A. Champine, director, advanced systems, for large scale commercial computer systems; and Dr. Barry A. Borgerson, director, research and technology, Sperry Univac research.